Available online at www.sciencedirect.com
ScienceDirect
Future Generation Computer Systems 24 (2008) 371–389
FGCS FUTURE GENERATION COMPUTER SYSTEMS
www.elsevier.com/locate/fgcs

Gridification of collaborative audiovisual organizations through the MediaGrid framework

Bruno Volckaert*, Tim Wauters, Marc De Leenheer, Pieter Thysebaert, Filip De Turck, Bart Dhoedt, Piet Demeester

Department of Information Technology, Ghent University - IBBT - IMEC, Gaston Crommenlaan, 8 bus 201, 9050 Ghent, Belgium

Received 30 August 2006; received in revised form 19 June 2007; accepted 22 June 2007
Available online 13 July 2007

Abstract

In this paper, we discuss a use case for employing Grid technology in a media production/distribution environment. This type of environment is faced with the challenges of storing, retrieving and processing massive amounts of digital multimedia data in a reliable and highly performing manner, preventing Grid technology from being introduced in a straightforward way. We therefore propose the use of a MediaGrid framework, providing support for content/resource sharing and advanced collaborative working. This MediaGrid framework and its components are discussed in detail, and, since the proposed MediaGrid is tailored to the needs of media production/distribution companies, we introduce these companies' profiles along with their typical user-task work flows, media application characteristics and service requirements. In order to experiment with various resource and topology setups and be able to develop and evaluate scheduling and QoS management algorithms that are tuned to the needs of these companies, we have developed MediaNSG, an advanced MediaGrid simulator which extends the ns-2 network simulator. MediaNSG's inner workings are detailed and finally, some MediaNSG proof-of-concept simulations are presented.

Keywords: Grid computing; Digital media; Grid simulation; Network awareness

1. Introduction

The media industry has been confronted with an increasing complexity in both the technical domain and the business domain, much in the same way as other businesses. Up until now, a broadcaster was an umbrella organization for different kinds of in-house activities like media production, distribution and playout, etc. More and more however, business drivers such as cost reduction, added value management, partnerships, global sourcing and business componentisation are forcing these companies to become more flexible, find partnerships (allowing computational and data resource sharing) and evolve into dynamically extending organizations, with business models based on business services available within the media market. These parameters combined with possible future mergers, acquisitions and fusions drive the media production/distribution business to become more agile.

Furthermore, exponential decrease in hard disk costs [1] ignited a paradigm shift in the production of audiovisual media from tape to file based. Current cost per byte of hard disk based storage systems rivals that of tape based systems and is expected to go below the stagnating prices of the latter [2,3]. Although today's architectures promise democratization of data access, i.e. inexpensive, non-mediated, and shared access to centrally-managed storage, this promise is only partially met by existing installations. On a software level, generic (Grid-enabled) applications are tuned towards typical ICT (i.e. Information & Communication Technology) related requirements and are currently not fit for the specific challenges induced by a file based media production and archiving platform.

In the long run, we generally expect the need for automated interaction between several audio/video media production sites, and the sharing of centralized storage, computational and specialized (e.g. capturing devices, broadcasting equipments) resources with several independent corporate users in a controlled manner. It is in this domain that media production

* Corresponding address: University Ghent, INTEC - IBCN - IBBT, Gaston Crommenlaan, 8 bus 201, 9050 Ghent, O-Vlaanderen, Belgium.
E-mail address: Bruno.Volckaert@intec.UGent.be (B. Volckaert).

0167-739X/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2007.06.009

environments can benefit from Grid technology to both improve media handling/processing times and provide the means for securely sharing and utilising distributed resources and applications amongst multiple virtual organizations by employing specialized Grid middleware.

Due to the specific scenario however, current Grid technology cannot be introduced in a straightforward way. The high bandwidth, reliability and short response time requirements when handling audio/video streams imply the need for special care in the design of the overall architecture and in particular in the scheduling, resource control and overall Grid management process. Media handling can take place at local sites before streaming them to a remote site or can be performed at a remote site: the scheduling, resource control and Quality of Service (QoS) management components of the Grid will have a high impact on the achieved application performance. Furthermore, the software architecture of the management platform will need to exhibit high performance and reliability to meet the specific application requirements.

The MediaGrid framework presented in this paper has been developed to cope with these challenges, and will make it possible for media partners to evolve into extended organizations where partnerships, media communities and commercialization of media services are omnipresent. Tasks submitted to a MediaGrid are treated according to the task, user and media company *profile* they originated from, allowing for QoS differentiation in the scheduling and resource control process.

Advantages of Grid-enabling the audiovisual media production/distribution companies would be:

- Ability to distribute media files among different companies within an environment with high reactivity requirements and various levels of Quality of Service
- Ease the exchange of media resources/assets (rendering farms, specialized media capture devices, etc.)
- Integration of broadcast media exchange standards (e.g. the EBU's P/Meta standard [4]) in a grid services environment to provide interoperability between different media content providers
- Migration from special purpose resources and applications to conventional IT hard/software thereby significantly lowering necessary investments
- Stimulate the growth of media community Virtual Organization (VO) setups supporting advanced collaborative working.

As a first step in order to test the MediaGrid framework for feasibility, and to allow the evaluation of MediaGrid-tuned scheduling and management algorithms without the need for time-consuming physical testbed setups, we developed MediaNSG, a MediaGrid simulator built on top of the ns-2 [19] network simulator.

This paper continues as follows: first we give an overview of the related works in Section 2, and continue by discussing the MediaGrid (Micro/MacroGrid) architecture in Section 3. MediaNSG, our MediaGrid simulator is discussed thoroughly in Section 4 followed by an overview of the different media production/distribution company profiles and the typical characteristics of their associated job classes in Section 5. Actual MediaNSG operation is detailed in Section 6, while some proof-of-concept simulation results are shown in Section 7. Finally, we give some concluding remarks in Section 8.

2. Related works

The Globus [5] toolkit, the open-source middleware for building Grids is the de facto standard Grid toolkit. The Globus toolkit is not an out-of-the-box solution however, it offers software services and libraries for resource monitoring, discovery, and management, plus security and file management, but these tools have to be deployed in a framework tuned to the Grid needs at hand.

GridCast [6,7] is a research project being undertaken by the British Broadcasting Corporation (BBC) and the Belfast e-Science Center aiming to develop a prototype media Grid, running on Globus middleware, that will manage the sharing of program content between distributed sites. The objectives are to effectively manage the distribution of broadcast media files, permit distributed processing and provide security and network resilience within a highly reactive environment requiring high levels of Quality of Service (QoS). The GridCast project focuses on the specific BBC topology (with regional BBC departments interacting with the main BBC production house), whereas we (in cooperation with the Flemish Radio and Television corporation—VRT [8]) aim at providing a general framework and focus on the accurate simulation of a multitude of collaboration setups between audiovisual companies.

Another project focusing on the MediaGrid concept is MediaGrid.org [9], which is a consortium looking into ways for computational grid platforms to provide digital media delivery, storage and processing services for networked applications. The envisaged media Grid combines Quality of Service and broadcast features with distributed parallel processing capabilities. The work contained in this paper is based on and an extension to some of the research works performed in the FIPA project [10]. The FIPA project (File based Integrated Production Architecture), is an IBBT (Interdisciplinary institute for BroadBand Technology [11]) project aiming at the development of an IP based architecture to share storage and computing power on single or multiple sites. Application areas are digital media production, e-security, e-health, etc. The FIPA project has successfully been ended in December 2006, and a follow-up project, the Geisha project [12] has already started (together with partners like a.o. IBM and VRT), focusing on the actual implementation of a service oriented architecture for MediaGrids. Currently, we are in the process of exchanging research results with the aforementioned MediaGrid.org consortium.

Some select media production hardware/software vendors are offering proprietary retail products hinting at distributed/Grid like behavior: A scalable solution for digital media post production networks is offered by Force10 Networks [13]. They mainly focus on the interconnectivity of rendering farms with several hundreds of cluster nodes through

Ethernet LANs. SGI [14] is known to be able to deploy an IT storage, computing and networking *hardware* infrastructure tuned to broadcast media environments. Another key player in the media distribution market is Anystream's Agility [15], which is a server based software used by content providers to streamline internal production and media exchange as well as to rapidly repurpose content for the Web, VOD, mobile IPTV and other emerging content outlets. None of these firms offer a full-featured hardware/software MediaGrid solution however.

Also, there are already full-featured storage solutions for digital media production/distribution companies: IBM's digital media center provides a shared, highly scalable, open-standard storage pool for broadcast production [16] and Omneon MediaGrid [17] offers a Grid-enabled content storage system that offers shared storage for all users within a facility and that is capable of performing media processing functions on stored content.

When we compare the MediaGrid concept to other current Grid use cases (as described in [18]), we notice similarities with the 'Commercial Data Center', 'On-line Media and Entertainment' and 'Inter Grid' use cases (the other use cases described in the referenced document are either too atomic or not applicable for the MediaGrid concept). The aspects which set the MediaGrid use case apart from the aforementioned use cases are the combination of, on one hand the inherent support for transport, storage, replication and processing (rendering, transcoding, etc.) of digital media (including metadata handling), on the other hand the mandatory QoS support (one simply cannot use best-effort services in a live media distribution environment) and finally the collaborative setup between multiple, geographically distributed audiovisual production facilities.

Accurate simulation of Grid systems is very important in order to be able to develop and evaluate Grid scheduling and management algorithms that are tuned to the actual situation. Simulation allows one to construct and test different Grid topology and Virtual Organization setups in a controlled environment, without the need for time-consuming physical construction of said configurations. This is also true in the case of MediaGrid systems, which is why we have developed MediaNSG, a MediaGrid simulator detailed in this paper, that was developed on top of the ns-2 [19] network simulator, and that can be seen as a network aware Grid simulator. While not providing the most scalable simulation kernel (more scalable C++ simulation frameworks are available, such as DaSSF [20] and OMNeT++ [21]), ns-2 is an up-to-date, discrete-event network simulator mostly used in academic networking research, partly due to its easy extendability (open-source with a large support community). Ns-2 provides models for a wide range of protocols for both wired and wireless networks. For media applications a slew of data transport and data streaming protocols are widely used. Most notable are the Real-Time Streaming Protocol (RTSP [22]), the Real-Time Transport Protocol and associated Real-Time Transport Control Protocol (RTP and RTCP [23]). Also resource reservation protocols like RSVP [24] can play an important role in supplying QoS to a MediaGrid. It is important to note that all these protocols are readily available in MediaNSG for simulating data streaming/transport and/or network resource reservation in MediaGrids.

Other notable existing Grid simulators include Bricks, MicroGrid, SimGrid, GridSim, ChicSim and OptorSim.

The Bricks Simulator [25,26] focuses on client–server interaction in global high performance computing systems. It allows for a single centralized scheduling strategy, which does not scale well with large Grid systems and does not support the notion of multiple (competing) schedulers.

MicroGrid [27] is an emulator modeled after Globus [5], allowing for the execution of Globus-enabled applications on a virtual Grid system. Research into the area of Grid scheduling algorithms can be cumbersome with this kind of approach, since it requires the construction of an actual Globus application to test.

SimGrid [28] is designed to simulate task scheduling (centralized or distributed) on Grids. Version 1 of SimGrid can be regarded as a low-level toolkit (which interfaces to the C programming language) from which domain-specific simulators can be built. The second version of SimGrid is dubbed MetaSimGrid [29] and is essentially a simulator built upon this toolkit to enable the construction of simulations with multiple schedulers (as C programs). Models for network links as well as for TCP connections are present in SimGrid. This validated TCP implementation allows for smaller simulation times when compared to the packet-level TCP simulation performed by network simulators. Of course, simulations using other transport protocols that are not readily available in SimGrid require that these protocols are implemented first, whereas using a network simulator ensures easy access to a wide range of protocols. The simulated application consists of several tasks, organized into a Directed Acyclic Graph (DAG). MetaSimGrid is focused on scheduling this application type in a master–slave environment.

GridSim [30] is a discrete-event Grid simulator based on JavaSim [31] (which has recently evolved into J-Sim [32] and has a similar Tcl/Java dual-language simulation environment as ns-2). This simulator allows to simulate distributed schedulers, and is specifically aimed at simulating market-driven economic resource models. While its computational resource models are highly configurable, only a basic notion of network connectivity is supported and underlying network dynamics are not simulated accurately.

The Chicago Simulator [33] is a simulation framework built on top of Parsec [34] for studying scheduling and replication strategies in Grids. A Grid is modeled as a collection of interconnected Grid sites with network connectivity of each Grid site modeled as a single parameter (describing the bandwidth of the gateway connecting this Grid site to the other Grid sites). As such, it does not provide the level of network resource detail that is modeled in MediaNSG.

OptorSim [35] is a Java based Grid simulator focusing on evaluating the performance of data access optimization algorithms. Its architecture is based on the EU DataGrid [36] architecture. OptorSim includes an economic model, using a peer-to-peer auction protocol that optimizes both the selection

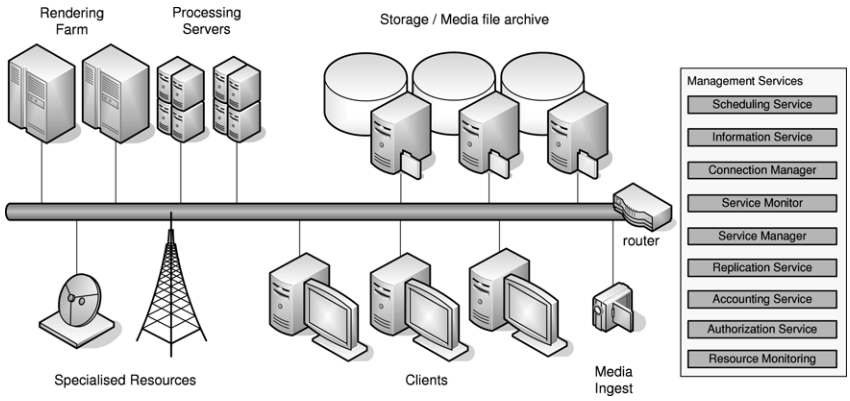


Fig. 1. Typical MicroGrid scenario.

Table 1
Grid simulator characteristics

	Scheduler	Network model	Packet-level simulation	Generic
Bricks	Single	Basic	No	Client–server
MicroGrid	Distributed	Very advanced	Yes	Globus emulator
SimGrid	Distributed	Advanced	No	Generic Grid
GridSim	Distributed	Basic	No	Generic Grid
ChicagoSim	Distributed	Basic	No	Generic Grid
OptorSim	Distributed	Basic	No	Generic Grid
MediaNSG	Distributed	Very advanced	Yes	Generic Grid

of replicas for running jobs and the dynamic creation of replicas in Grid sites using a file revenue prediction function. OptorSim takes network bandwidth into account when transferring the job input/output data (although it does not actually simulate any existing network protocols) and currently has no notion of Grid services.

Table 1 summarizes the main differences between the discussed Grid simulators.

3. MediaGrid framework

The MediaGrid framework is built on the MicroGrid concept on one side and the MacroGrid concept on the other. In what follows we will explain these two constituents in detail.

3.1. MicroGrid details

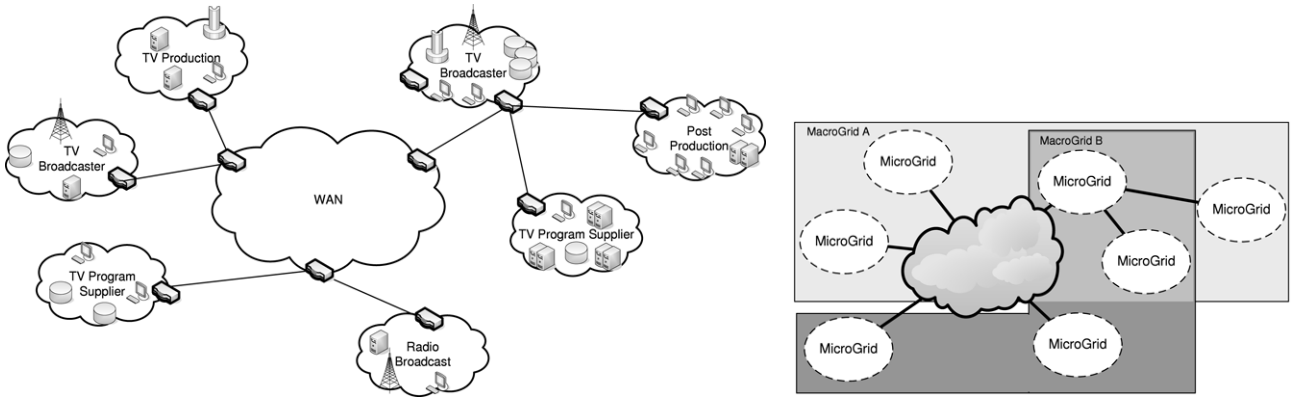
A MicroGrid (see Fig. 1) denotes a Grid setup at a local audiovisual media production and/or distribution facility, interconnecting the different local resources and providing the tools to manage, access and control local (self-owned) resources and digital media archives. Resources can be storage/data resources, providing disk space for storing and retrieving media files respectively, or computational resources, which in turn provide the computational power required for processing the different user submitted tasks. In a media production company, one typically discerns computational resources located in terminals (with a high degree of interaction between the job and the user e.g. editing terminals) and computational resource farms (focused on fast processing of computationally intensive tasks e.g. rendering or transcoding). The MicroGrid can also allow Grid access to specialized resources (capturing devices, broadcasting equipment, etc.).

Each MicroGrid needs a set of Grid management components to be able to tackle issues such as job scheduling, Quality of Service, etc. Required management components are: a scheduling system (responsible for the 'intelligent' allocation of resources to jobs according to a certain objective e.g. minimize job turnaround time or meet specific deadlines), information service (storing registered MicroGrid resource's properties and characteristics), monitoring system (monitoring the status of computational, storage and data resources), connection manager (responsible for monitoring the status of network resources and setting up network connection reservations), service monitor (monitoring QoS requirements of jobs and collecting service class information) and a service manager (which reserves resources for service classes in order to provide them with specific QoS guarantees). Other notable management components include an accounting component, an authorization/security component (for restricting MicroGrid resource access) and a data transfer/replica manager (responsible for replicating/caching frequently accessed media files).

A MicroGrid can thus be seen as a provider of a set of Grid services, and these services can be advertised not only to the local company users, but if wanted also to 3rd party media companies with which one wants to collaborate (see Section 3.2). Each offered service is accompanied by extensive access control (based on Grid certificates and directory services describing which user/user class can use a particular service and to what extent) and accounting agreements (e.g. service usage pricing information for different user classes), allowing MicroGrid managers full control over how local resources may be utilised by *MediaGrid* users.

3.2. MacroGrid details

A MacroGrid is a collection of interconnected MicroGrid sites. In such a MacroGrid, resources can be shared amongst the different constituent MicroGrids (while taking into account the access policies of each MicroGrid's resource usage service).



(a) Physical MacroGrid.

(b) Logical MacroGrid.

Fig. 2. Physical and logical MacroGrid setups.

This way, jobs that originate at one MicroGrid site, can be migrated to another MicroGrid for processing (e.g. in case insufficient processing power is available at the originating site or if the job needs to have access to specialized resources not available at its originating site). The MicroGrid schedulers query the different MicroGrid sites' information and monitoring services for resources adhering to a user job's requirements and decide whether it is beneficial (e.g. faster processing times) or necessary (e.g. specialized resources unavailable locally or when having to cope with local resource failures) to utilise remote resources, while taking into account the possible downsides of using remote resources (price tag, larger network transfer times due to remote location, etc.).

Besides resource sharing, MacroGrids also enable media file sharing between different MicroGrids. In this case, users are able to utilise/retrieve/store remote media files, again taking into account any access restrictions that have been specified (speed/ratio limitations in order not to deteriorate internal MicroGrid performance, clearance levels depending on the content/copyrights of media files, etc.).

Another important benefit of sharing MicroGrid services across company boundaries is the ability for users to work on projects collaboratively. The accounting managers of each MicroGrid can be used to keep track of resource and media file service usage by users/MicroGrids allowing economic gains by charging money and/or bartering for external resource usage compensation.

It is important to note that a MacroGrid does not have exclusive access to a MicroGrid's services: one MicroGrid can be included in multiple MacroGrids (and each membership can come with different resource/media file usage policies and access control configurations), with each MacroGrid representing a different Virtual Organization (see Fig. 2). Also, MacroGrids are not necessarily static structures, in which MicroGrids can join or leave this Virtual Organization at any time by changing service access policies.

4. MediaGrid simulation

If we wish to develop scheduling/service management algorithms tuned to the requirements of MediaGrids, or wish to evaluate the performance of different network/computational/storage resource configurations, we either have to construct a testbed and measure task/resource performance, or we can resort to accurately simulating a MediaGrid's behavior. Due to the size and amount of resources involved in setting up a realistic MediaGrid testbed each time a new scenario needs to be evaluated, accurate simulation of MediaGrid scenarios is likely to be more efficient.

MediaNSG is a network aware Grid simulator built on top of ns-2 [19] and has been developed to allow users to simulate typical task submission behavior of different digital media company organizations and experiment with scheduling and service management architectures. MediaNSG supports the simulation of both MicroGrid and MacroGrid behavior, and provides the user with output data regarding job execution statistics (job response time, time spent in scheduling queue, data transfer size/speed, etc.) for the different tasks, resource (computational, storage and network resources) and management component (scheduler, information service, etc.), usage statistics, bottlenecks, etc.

From an architectural point of view, MediaNSG is comprised of two distinct layers: a Grid layer in which the different management components, resources and jobs are modeled, and a network layer through which all ns-2 functionality is available (see Fig. 3). Each component in the Grid layer is mapped to a node in the network layer, while nodes themselves can be interconnected through a wide variety of wired and wireless network technologies and protocols (more specifically: all network technologies and protocols that are present in ns-2). If a Grid component/resource needs to send data to another component/resource, the actual data transfer is fully simulated (up to packet level), allowing for accurate network transfer time to be employed in all simulations. We use a generic and portable XML-format for intercomponent control message exchanges.

The MediaNSG Grid simulation layer C++ source is composed of 47 classes (not including topology generation/GUI visualization tools), comprising 19.000 lines of code (19 KLOC). The most important classes are shown in Fig. 4. All elements that can be assigned to an ns-2 node (resources, management components and clients) inherit from the *GridObject* class,

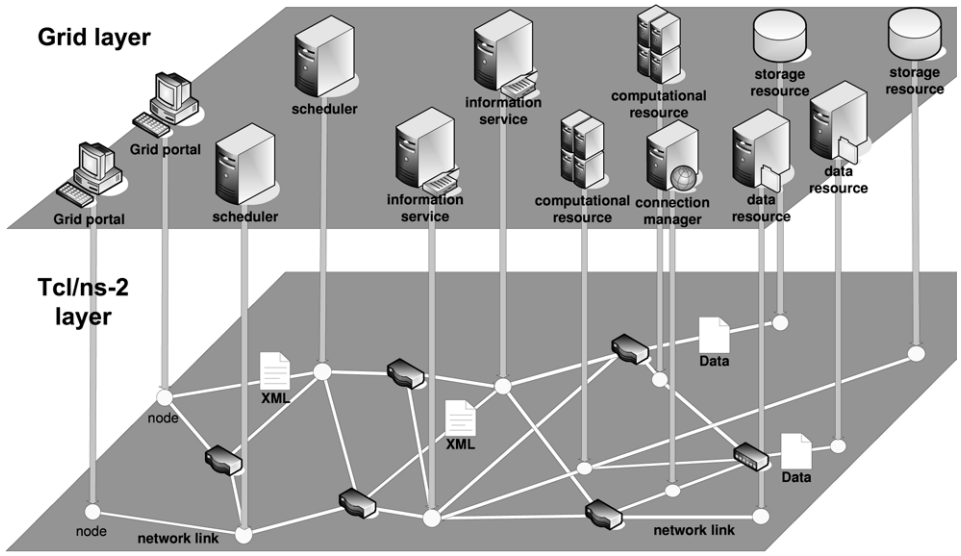


Fig. 3. MediaNSG simulator architecture.

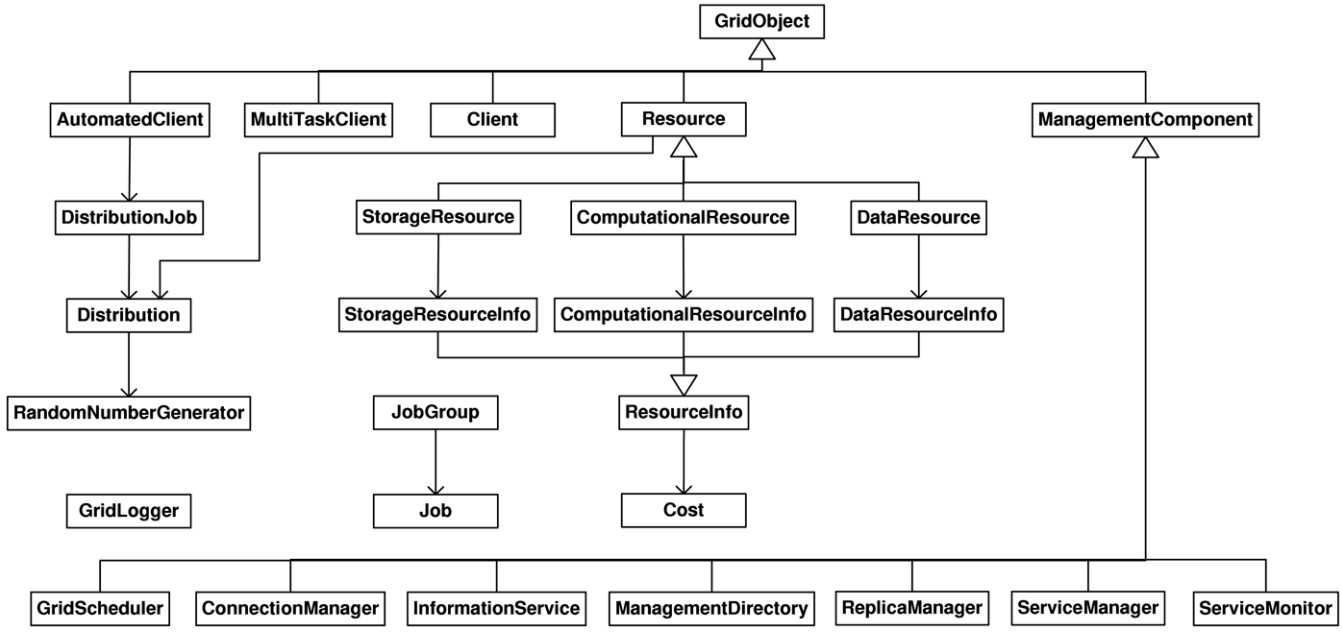
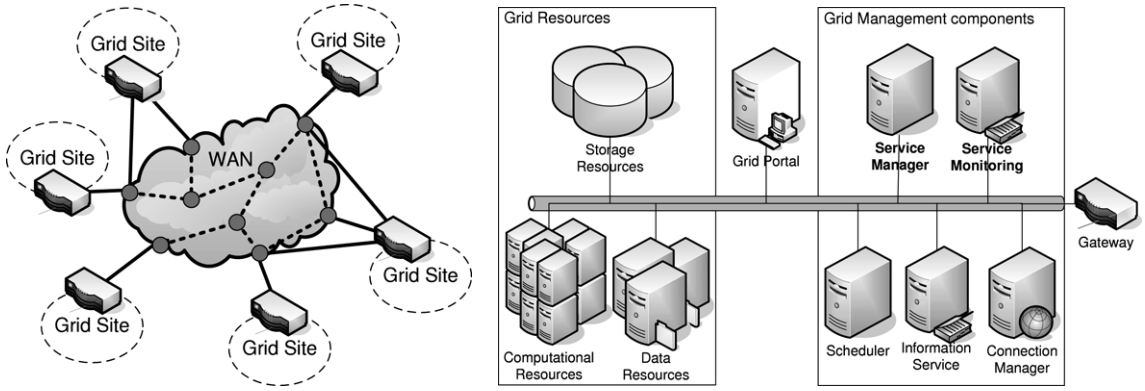


Fig. 4. MediaNSG implementation architecture.

which connects the Grid simulation layer with the ns-2 network simulation layer by providing methods for sending/receiving data and event control. Each *Resource* object contains a *ResourceInfo* object which stores resource properties/status information and provides methods for reading/writing from/to XML (for sending resource information between management components). The *GridLogger* class provides simulation logging functionality with support for multiple log levels (MediaNSG users can select the logging levels they want to see messages from) and error reporting.

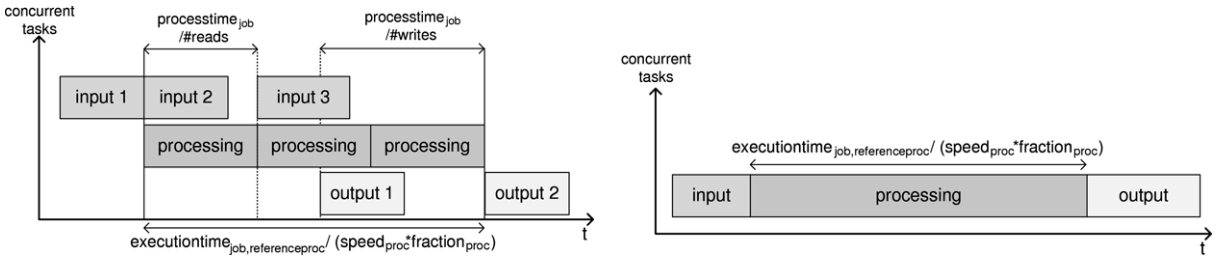
4.1. Grid model

In MediaNSG, we regard a MacroGrid as a collection of *MicroGrid sites* interconnected by WAN and MAN links (see Fig. 5). Each MicroGrid site has its own resources (computational, storage and data resources) and a set of management components, all of which are interconnected by means of LAN links. Management components include a *Connection Manager* (capable of offering network QoS by providing bandwidth reservation support, and responsible for monitoring available link bandwidth and delay), an *Information Service* (storing registered resources' properties and monitoring their status), a *Scheduler*, a *Service manager*, a *Service monitor* (these deliver advance resource reservation support in order to provide Quality of Service to the jobs) and a *Replica Manager*. The explicit treatment of the network as a "resource" allows management components to take decisions based on observed and expected future load of the network.



(a) MacroGrid. (b) MicroGrid site.

Fig. 5. MediaNSG Grid model.



(a) Non-stalling job, simultaneous transfer and execution. (b) Non-stalling job, pre-staged input data.

Fig. 6. MediaNSG job/application model.

Each MicroGrid site can have one or more Grid portals, through which clients can submit the jobs. These jobs are then scheduled on a collection of resources by a Scheduler. To this end, the Scheduler makes reservations with the resource managers; in our environment, a Connection Manager manages a collection of network links, while the Computational, Storage and Data resources double as their own manager. To ensure connectivity with the outside world (and in particular with other MicroGrid sites), each MicroGrid site designates one or more of its underlying ns-2 network nodes as a gateway to the WAN/MAN.

4.2. Job model

The basic unit of work in our model is a *job*, which can roughly be characterized by its duration (time it takes to execute on a reference processor), computational requirements (memory, operating system, installed applications, etc.), the required input data, the output data size, the *burstiness* with which these data streams are read or written, and the service class to which it belongs. Knowing the job's total duration and the frequency at which each input (output) stream is read (written), the total execution duration of a job can be seen as a concatenation of instruction "blocks". The block of input data to be processed in such an instruction block is to be present before the start of the instruction block; that data is therefore transferred from the input source at the start of the previous instruction block. Similarly, the output data produced by each instruction block is sent out at the beginning of the next instruction block. We assume these input and output transfers occur in parallel with the execution of an instruction block. Only when input data is not available at the beginning of an instruction block or previous output data has not been completely transferred yet, a job is suspended until the operation that causes the stalling completes. A typical job execution cycle (one input stream and one output stream) is shown in Fig. 6. The presented model allows us to mimic both data *streaming* (high read or write frequency) and *data staging* approaches (read frequency set to 1).

An overview of all MediaNSG job parameters is given by:

- Job arrival time at MicroGrid portal
- MicroGrid and user from which the job originated
- Start time: time at which the job may begin processing
- Amount of processing time needed on reference processor
- Duration: minimum duration of the job (to ensure that job processing occurs at a defined rate e.g. video viewing jobs)
- Minimum processor speed asked for by the job
- Temporary disk space required on processing element
- Software installation (operating system/programs) required for the job to run
- Input data: access type (GridFTP, secure copy, etc.), minimum required retrieval speed, ID of input sets required, number of reads
- Output data: access type (GridFTP, secure copy, etc.), minimum required storage speed, number of writes, time output data needs to be kept available
- Budget: maximum cost of executing the job
- Deadline
- Service class (denotes application from which the job is spawned).

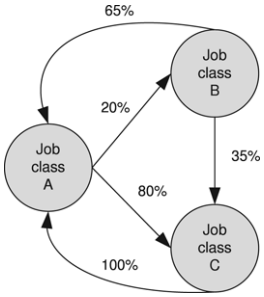


Fig. 7. Sample multi-service client workflow.

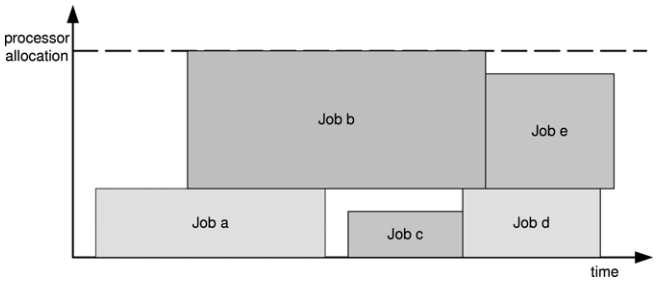


Fig. 8. Computational Resource processor allocation.

4.3. Client model

In our simulation environment, a Client is a component which automatically submits the jobs from a particular service class to a Grid portal (which in turn delivers these jobs to a Scheduler). The "home site" of these jobs is the MicroGrid site where the Client is located. All the job characteristics, together with the job submission start/end times and the job interarrival times are specified statistically (normal, uniform, zipf and exponential distributions are supported) according to the Client's configuration (specified in a MediaNSG Tcl startup script) or, alternatively, are read from trace files containing previously recorded job submission behavior. The latter approach allows for identical job load reproduction under different MediaGrid topologies/resource setups. Note that Clients are not required to wait for a previously submitted job to finish before launching another job.

Multi-service class Clients, capable of constructing and submitting jobs from different service classes (with job parameters generated from distinct service class' configurations) can be instantiated in MediaNSG. Each multi-service Client can have a service class work flow assigned to them. These work flows define the probability x of constructing and submitting a job from service class y when a job from service class z was last submitted by this client (see Fig. 7).

Clients are specified by:

- A list of job/service classes with job parameters specified statistically
- Work flow between the different job/service classes
- Grid portal location
- Job generation start time
- Job generation end time.

4.4. Resource models

4.4.1. Computational Resource model

Each Computational Resource is viewed as a monolithic entity with a certain processing power. Its main capabilities are defined by the following parameters:

- The number of processors and their respective processing power (relative to a reference processor)
- Memory available to the jobs
- Disk space available for storing temporary job output
- Installed operating system and applications/software
- Load: job load (processing, memory, disk space) and reservations
- Dynamic resource model: resource failure probability, probability to go off-line and average time before the resource restarts
- Cost: price when using this computational resource (depending on user class of client that submitted the job to be processed).

This model can be used to represent both multiprocessors and clusters, provided that, in the latter case, the internal network connecting the various cluster nodes performs sufficiently (in a well balanced multiprocessor system, the network bus interconnecting the processors will only rarely be a performance-limiting bottleneck). If the network interconnecting the different processing elements can be the source of bottlenecks, one should instantiate a Computational Resource for each processing element and interconnect these resources by means of bandwidth limited network resources (see Section 4.4.4).

Before accepting a job for processing, a Computational Resource will check the requirements of the computational reservation (sent by a Scheduler) to see if they do not conflict with existing and/or future computational reservations. If there is a conflict, the Computational Resource will reject the reservation and inform the requesting Scheduler. Once a job is accepted for processing, the job description is parsed for information regarding (optional) Data and Storage Resources that are to be used for retrieving/storing input/output data. In our model, the Computational Resource is responsible for starting the retrieval of each input block and sending output blocks to the necessary Storage Resources.

As can be seen in Fig. 8, a computational reservation allocates a fixed fraction of the Computational Resource's processing power over a certain amount of time (so it can ensure that deadlines will be observed). During the lifespan of the reservation, the allocated fraction itself will never be modified, but, due to job blocking, the time this processor fraction is allocated to a job can be enlarged if it does not infringe other computational reservations.

When a job has finished sending its last output block (or, if no output had to be sent, once the job has no further processing to do) the resource will inform the responsible Scheduler that the job is finished, and will release the job's computational reservation.

4.4.2. Storage Resource model

Storage Resources provide disk space to store the job output data. In our model, Storage Resources are described by

- The total available storage space
- Load: storage space allocated to jobs and stored media
- Dynamic resource model
- Cost: price when using the storage resource (depending on user class of the client that submitted the job).

When a Storage Resource receives a storage reservation request, it checks to see if this reservation does not conflict with the already granted reservations (in terms of disk space) and, if possible, grants the reservation. A job can choose to keep its output data stored until the job finishes processing (at which time the Storage Resource will release the job's storage reservation), or it can opt to keep output data stored for a specified time (at the possible expense of extra cost).

4.4.3. Data Resource model

Data Resources serve the purpose of providing input data for the jobs. In our MediaNSG model, Data Resources are described by

- Available datasets (by ID), their respective sizes and usage characteristics (the latter is for replication purposes)
- Available storage space for datasets
- Load: content being read by jobs and reservations
- Dynamic resource model
- Cost: price when using the Data Resource (depending on user class of the client that submitted the job).

Each time a job retrieves an input dataset, the Data Resource updates its internal dataset usage properties (time accessed, number of times the dataset has been accessed in last time frame). These usage characteristics can in turn be used to decide which datasets no longer will be supported in favor of new (replicated) datasets (i.e. storage space will be made available for storing often utilised, replicated datasets by freeing disk space taken up by less frequently used datasets; for more information see Section 4.5.2).

4.4.4. Network Resource model

Interconnections between resources (i.e. between two non-network resources) are modeled as a set of network links, providing a route between the source and destination resource. Connection reservations, each offering a guaranteed total bandwidth available to a particular Grid job or service class, can be setup by the Connection Manager. Of course, these connections can only be setup if, in the underlying ns-2 network topology, a route (with sufficient bandwidth capacity) exists between the nodes to which these resources are attached. Grid resources can also be interconnected by means of reserved network tunnels (e.g. VPN tunnel). In this case, a tunnel (with guaranteed bandwidth availability) is setup between Grid resources for a particular Grid job service class (see Fig. 9).

Connections in our model are a set of network links, with each network link described by:

- Endpoints
- Total bandwidth capacity

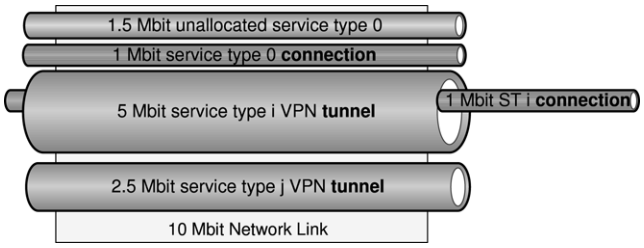


Fig. 9. Network model.

- Residual bandwidth of non-tunneled connections over time (advance network reservations are supported)
- Residual bandwidth for each pre-assigned tunnel over time (advance network reservations are supported)
- Delay
- Connection Manager responsible for network link (see Section 4.5.3).

4.5. Management components

4.5.1. Information Service

An Information Service offers a Computational, Storage and Data Resources' property and status repository. Each time a resource is instantiated in the Grid, it will register its static properties and dynamic status info with at least one Information Service. An Information Service can be queried by any other Grid management components (e.g. Scheduler, Service Manager) for resources meeting certain requirements (e.g. available memory, installed software, available datasets). The Information Services perform matchmaking on this query, i.e. they select a set of resources (from their registered resource repository) that meet the requirements of the resource query. The resulting resource set is sent back to the management component requesting resource information. Typically, since a single central information service for a Grid would not be scalable, each MicroGrid site offers one (or more) Information Service component storing (a subset of) the local resources.

Each time a resource's state changes (e.g. because of accepting a new job for processing) it contacts the Information Services it is registered with and sends up-to-date status information (note that these status update messages, as any control message in MediaNSG, can suffer from network delays, temporarily rendering the status information contained within the Information Service outdated).

The Information Service also monitors the liveliness of each registered resource (in order to detect resources failures). At configurable intervals it sends a heartbeat message to these resources and expects to receive the responses in adequate time (see Section 4.7). If these responses do not arrive in due time, the resource is unregistered from the repository.

4.5.2. Replica manager

A Replica Manager monitors the job input data retrieval behavior in such a way that when a job is in need of a dataset that is not present at the job's computational processing site (which is not necessarily the job's originating site), a replica note is sent to that MicroGrid site's local Data Resources,

Table 2
Audiovisual applications: average network, processing, storage and QoS requirements

No		Bandwidth	CPU	Storage (GB/h)	QoS
1	Ingest	Lo- or HiRes A/V	Low	0.65–25.7	High
2	Quality checking, HiRes browse	HiRes A/V	Low	25.7	Low
3	LoRes browse	LoRes A/V	Medium	0.65	Low
4	LoRes rough EDL	LoRes V, Lo- or HiRes A	High	0.5; 0.15–0.7	Medium
5	Send/restore archive	Lo- or HiRes A/V	Low	0.65–25.7	Medium
6	Craft editing	5–10 HiRes A/V	High	5–10 ∗ 25.7	High
7	Rendering, conforming, transcoding	HiRes A/V	High	25.7	Low
8	Playout	1–40 HiRes A/V	Low	1–40 ∗ 25.7	High
9	Audio editing	Lo- or HiRes A/V	High	0.65–25.7	Medium
10	Graphic creation	HiRes V	High	25	Low

asking them if they are interested in replicating that dataset locally (the job's needed dataset will be transferred from a remote Data Resource to the Computational Resource where the job is processing, so it will be available for replication). The Data Resources can then choose to either ignore this replication request (based on the usage characteristics of their offered datasets) or they can store the new dataset locally. If a Data Resource decides to replicate the data locally, it can simple store the dataset along with the existing datasets (if there is enough free storage space) or it can choose to replace an existing dataset using a Least Recently Used algorithm (in this case the least recently used dataset(s) will no longer be supported from that particular data resource).

4.5.3. Connection Manager

A Connection Manager monitors properties (e.g. total bandwidth) and status information (e.g. available bandwidth, delay) of the different Grid network links. It can be queried by any other management component to retrieve connection information (available bandwidth, delay, routing information, total bandwidth, etc.) between a source and destination network node. If a connection query is received, the Connection Manager will ask ns-2 for the route from source to destination and, once routing information has been received, will inspect the status of all network links along this route. The Connection Manager will then construct a connection information object, containing info about all links along this route and available bandwidth/delay between source and destination. This connection information will be sent back (in XML) to the querying management component.

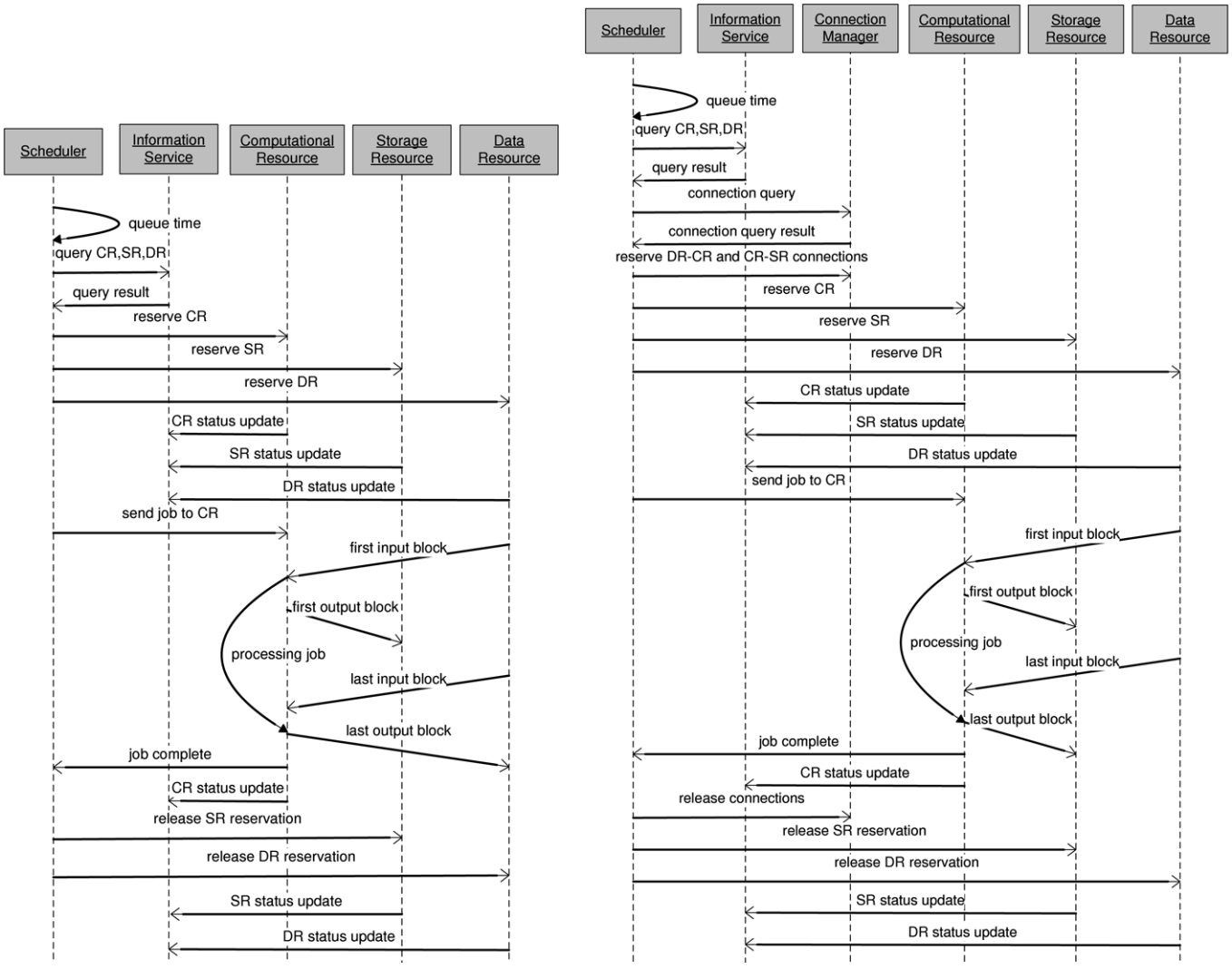
Next to an informative function, a Connection Manager is also responsible for setting up network resource reservations. Both end-to-end connections and service class tunnel reservations are supported (see network model in Section 4.4.4). To this end, management components can ask the Connection Manager to reserve a certain amount of bandwidth between a source and destination node. The Connection Manager will inspect all network links on the route between source and destination, and, if bandwidth requirements are met on each link, will reserve the requested amount of bandwidth, grant the connection reservation and inform the requesting management component.

Note that reservations are not physically setup by the Connection Manager: if the bandwidth requirements of the requested connection reservation are not infringing previously guaranteed connection reservations' minimum bandwidth, the request is granted. If however this is not the case (due to the use of stale resource state information when assigning resources to jobs in the scheduling round), the connection reservation request is rejected and the job will be put back in the scheduler queue until the next scheduling round. The Connection Manager thus operates by bookkeeping all granted connection reservations and denying new reservations that would infringe on those previously granted reservations.

4.6. Scheduling

When the jobs are submitted, a Scheduler needs to decide where to place the job for execution. A Scheduler therefore queries the Information Services for resources that adhere to the requirements of the jobs it has to schedule. Once resource query results are in, the Scheduler employs one of its scheduling algorithms in order to assign resources to the jobs in its scheduling queue. The scheduling algorithm used in making this selection has a big impact on Grid performance, and influences overall Grid job throughput, Grid resource efficiency etc. If the Scheduler is unable to allocate the needed resources for a job, the job gets queued for reschedulement in the next scheduling round. The time between two scheduling rounds can be fixed, but it is also possible to set a threshold (e.g. time limit or number of unscheduled jobs in the queue) which triggers the next scheduling round. During each scheduling round, a scheduling algorithm processes submitted yet unscheduled jobs (note that the jobs currently residing in the scheduling queue can be reordered based on their QoS requirements as seen in Table 2—effectively making sure that the scheduler first tries to assign available resources to high QoS jobs), and attempts to minimize the completion time for each job. Once scheduled, our scheduler does not attempt to pre-empt jobs. MediaNSG has built-in support for multiple scheduling algorithms ranging from network unaware to network aware scheduling algorithms, priority based scheduling, least cost scheduling, etc. In the proof-of-concept simulations described in Section 7, two distinct scheduling heuristics were employed: non-network aware and network aware scheduling.

Non-network aware scheduling will compute Grid job schedules based on the status of the Computational, Storage and Data Resources. Algorithms that use this kind of approach



(a) Non-network aware job scheduling.

(b) Network aware job scheduling.

Fig. 10. MediaNSG scheduling model.

will not take into account information concerning the status of resource interconnections. The decision of which resources to use for a job will be based on the information acquired from the different Information Services (i.e. job execution speed and end time will be calculated based on the status of the available resources). Because non-network aware algorithms assume that residual bandwidth on network links is "sufficient", jobs can stall on I/O operations: their computational progress is no longer determined by the Computational Resource's processor fraction that has been allocated to it (which, together with the job's duration on a reference processor and the Computational Resource's relative speed determines its earliest end time *if all input and output transfers complete on time i.e. before the start of the appropriate instruction block*), but rather by the limited bandwidth available to its input and output streams. Note that the fact that network information is discarded during the scheduling implies that no connection reservations (providing guaranteed available bandwidths) are made—these would allow to accurately predict the job's running time.

If the scheduling algorithm is network aware (see Fig. 10), the Connection Manager is queried for information about available bandwidth on (shortest route) paths between resources and, once a scheduling decision is made (taking into account the speed at which I/O data can be fetched/stored to/from the processing job and adjusting computational power that gets reserved for this job to match), attempts to make connection reservations between the selected resources; connection reservations provide a guaranteed minimum bandwidth available for that job. Note that reservations are not physically setup by the Connection Manager: if the bandwidth requirements of the requested connection reservation are not infringing previously guaranteed connection reservations' minimum bandwidth, the request is granted. If however this is not the case, the connection reservation request is rejected and the job will be put back in the scheduler queue until the next scheduling round. Once all resource reservations are successful, the job is sent to the selected computational resource which takes care of fetching the different input datasets and storing the job's output data.

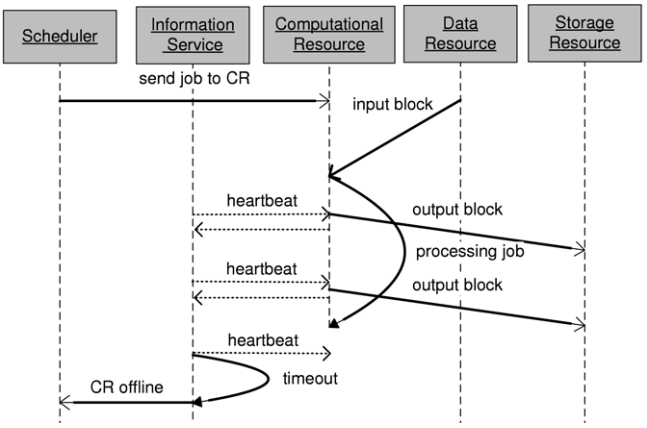


Fig. 11. Computational Resource failure.

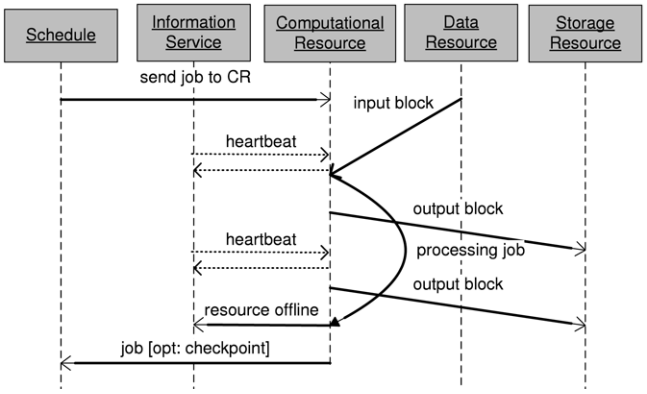


Fig. 12. Computational Resource unavailability.

4.7. Dynamic Resource model

Sudden failure of Grid resources (Computational, Data and Storage Resources) is detected by the Information Service components and the appropriate actions are taken to ensure that the jobs that were relying on the failing resource get rescheduled. Our dynamic model supports two notions of unavailability:

4.7.1. Resource failures

Unexpected Grid resource unavailability. These failures will be detected by the Information Service (who periodically sends a heartbeat message to each resource registered with it). If a resource does not reply to this message within a specified time-interval, failure is assumed. The Information Service proceeds to unregister the resource from its repository, and sends a notification message to the Grid Scheduler(s) that had jobs running on the crashed resource. The affected Schedulers then revert the jobs that were utilising the failing resource to the unscheduled state and put them back in the scheduling queue (see Fig. 11). Resource failures can be specified by means of two distribution-type parameters: “time before resource failure” and “time before resource restart”.

4.7.2. Resource unavailability

Each Grid resource may unregister itself at any time by sending a message to the Information Service it is registered with. The resource will then proceed by contacting the Grid Schedulers that have jobs utilising it (either for job processing or for retrieval/storage of I/O data). If checkpointing is enabled, the last checkpoint of jobs running on that particular resource will be sent to the Scheduler responsible for allocating the job’s resources (see Fig. 12). When a Scheduler is notified of the unavailability of a resource, it looks up which jobs were scheduled on that resource, cancels all resource reservations of those jobs, reverts the state of those jobs to their initial state (the “unscheduled” state with no work done) and readies the jobs for rescheduling. If a checkpoint is available, the job will continue processing from that checkpoint onwards.

5. MediaGrid profiles

5.1. MediaGrid application profiles

Together with partners from the media industry (more specifically the Flemish Radio and Television Network [8] and Video Promotion [37], a company active in the broadcast television market), we studied the characteristics and requirements for the audiovisual applications that are to be supported by the MediaGrid architecture. This resulted in task, user and company profiles that have been implemented in the MediaNSG simulator and that can readily be used for simulations.

Audiovisual application classes show large differences in their processing, network and storage requirements. In Table 2 we give an overview of average task class/application requirements of the most typical tasks/applications in a media centered company. From left to right we give average bandwidth requirements (A stands for audio, V for video and A/V for simultaneous audio/video streams; audio and video can be in either low resolution or high resolution streams depending on the application), processing requirements, storage requirements (in amount of storage that is needed per hour of A/V data). The Quality of Service parameter can be used by MediaNSG while scheduling and during service management to ensure that priorities are given to high QoS tasks. Table 3 shows the network and storage requirements for different resolution audio and video streams (e.g. one high resolution audio stream produces 1.5 Megabit per second of network traffic when streamed, and is in need of about 0.7 GB storage space per hour).

- Ingest: deals with bringing media files onto the storage/archive system, extracting keyframes and constructing metadata about the ingested media.
- Quality checking, HiRes Browse: tasks from this class inspect the quality of media files in high resolution to see if it is fit for playout.
- LoRes browse: mainly used to rapidly shuffle through different archived media files in low resolution when trying to find specific or suitable source material.
- LoRes rough EDL: construction of a rough Edit Decision List (EDL). This Edit Decision List is a list of events

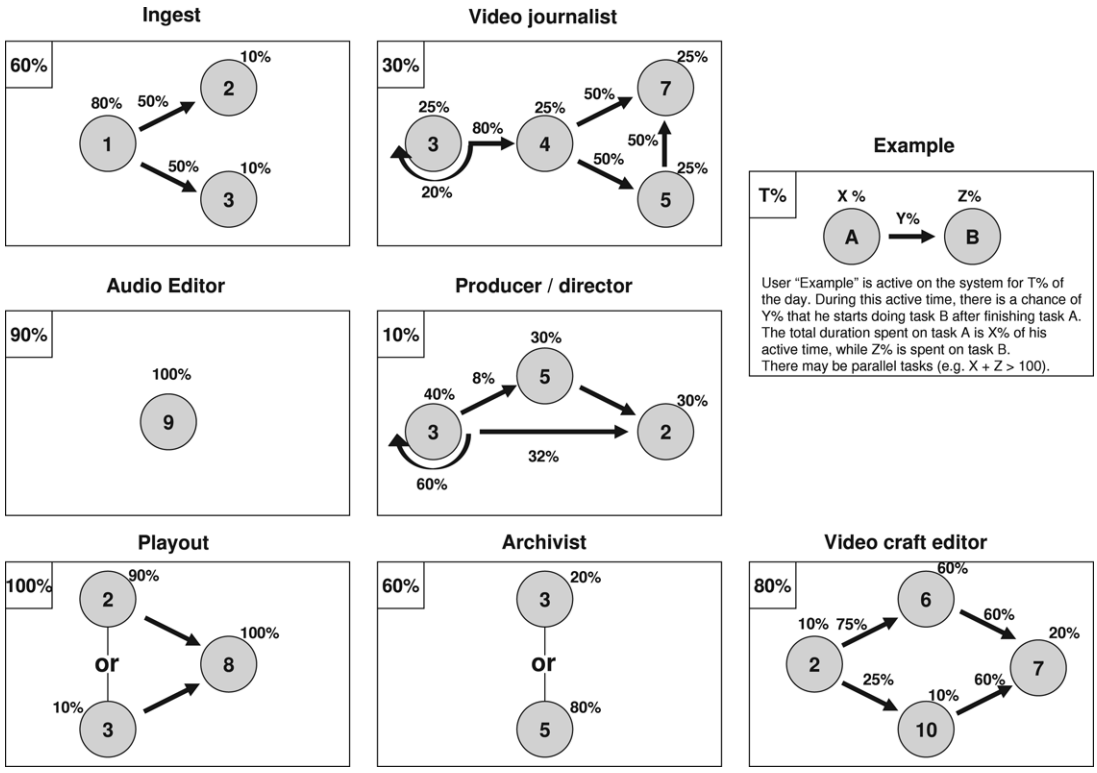


Fig. 13. Task workflow of typical audiovisual company user classes.

Table 3
Network and storage requirements of typical audio/video streams

Streams	Bitrates	Storage (GB/h)
HiRes video	20–50 Mb/s	25
LoRes video	1 Mb/s	0.5
HiRes audio	1.5 Mb/s	0.7
LoRes audio	256 kb/s	0.15
HD HiRes video	200 Mb/s	100

that include the sources to be recorded from and information about transitions (cuts, dissolves, wipes), transition durations, etc. Once an EDL has been processed, the result will be a newly constructed media file.

- Send to/Restore from archive: fetching data from the archive or storing new media files mainly stresses the available network resources.
- Craft editing: high quality finegrained editing and jog shuttling of multiple audio/video streams.
- Rendering, conforming, transcoding: this task involves rendering graphics, conforming of media to different video standards and transcoding of audio/video data to different qualities/resolutions/standards.
- Playout: Viewing multiple audio/video streams and sending one of those to playout equipment (e.g. broadcast equipment).
- Audio editing: Editing of multiple audio streams (possibly in conjunction with a video stream that needs to have the associated audio stream edited)
- Graphic creation: The creation of CGI imagery, custom scene transitions, etc.

5.2. MediaGrid user profiles

Each user belongs to a particular user class (with the latter describing the characteristics of job types that are to be submitted). Every time a user wishes to launch a job, it constructs a job from one of its user class's registered job types and waits until the job has been scheduled and processed by the MediaGrid. When the job is finished, the user class's work flow (as seen in Fig. 13) is inspected to determine from which application type a new job is to be generated. Note that the task numbers in these work flows refer to the numbering (first column) that is used in Table 2. We now look at the different user classes of typical audiovisual companies, with each user class showing widely differing characteristics depending on which applications they use:

- Ingester: This profile includes tasks like quality checking and low resolution browsing, besides the actual ingesting of media onto the storage archive.
- Video journalist: The main tasks of a journalist are low resolution browsing, low resolution rough EDL construction (Edit Decision List) and rendering, conforming and transcoding.
- Audio/Video editor: an audio editor deals with mixing and editing multiple audio tracks, while video editing includes quality checking, craft editing, rendering/conforming/transcoding and graphic creation.
- Producer/Director: involved at different stages of media production, mainly doing low resolution browse tasks, with the occasional sending to/restoring from archive and some quality checking and/or high resolution browsing.

Table 4
Audiovisual company average user class representation

	Ingest	Video journ.	Audio ed.	Video ed.	Prod./director	Playout	Archivist
Regional TV prod.	2	30–50	2–3			2	2
National TV prod.	3	300–500	20–30			3	4
TV post prod.	1	10–50	1–3	5–20	10	1	1
TV broadcast	1	5–10	1–5			1	1
TV program supplier			25		25		
Video on demand	2		5			2	1
Radio prod./broadcast			30		20	50	

- Playout: tasks include quality checking, low resolution browsing and playout.
- Archivist: an archivist mainly performs low resolution browsing and sending to/restoring from archive.

This information, together with the user/task work flows (presented in Fig. 13) and average application characteristics presented in Table 2, has been used to construct accurate media user profiles for use in MediaNSG simulations.

5.3. MediaGrid company profiles

Profiles have been provided for typical audiovisual companies (mainly describing the average amount of users from each user class working simultaneously as can be seen in Table 4). The most important profiles are:

- Television production: an example of television production is news program production. In these organizations tens (regional) or hundreds (national) of video journalists gather information that has to be ingested, edited, archived and played out.
- Television post production: in a post-production facility the same user classes are present, along with producers/directors managing the studio work.
- Television broadcast: television broadcast companies are not involved in (post-) production. The focus is more on playout than on editing.
- Television program supplier: these companies combine individual items into finished programs and send these to television broadcasters. Editors and producers/directors are the most important user classes in this type of organization.
- Video on Demand: companies delivering Video on Demand services mainly focus on indexing of the available material, user and channel dependent encoding of the streams and playout.
- Radio broadcast: similar to television broadcast, but with different requirements (e.g. no buffering or delays allowed).

6. MediaNSG operation

In order to setup a simulation, users must provide both a company and resource topology description. The default organizations that are discussed in Section 5.3, are all readily available through the MediaNSG frontend. Each organization is modeled as a collection of users belonging to different user types (see Section 5.2), with each user in turn being modeled as a task submitting entity (i.e. users submit tasks according to the task work flows discussed in Fig. 13). The default tasks described in Section 5.1, along with all their properties (CPU utilisation, storage needs, bandwidth, QoS, etc.) have been supplied, and all task characteristics can be modified through the GUI (see Fig. 14). New media organization profiles can be added, and existing profiles can be modified to include additional users and/or jobs.

Currently, simulated MicroGrid topologies deploy one central Storage/Data resource by default, with each client submitting jobs from a dedicated Computational Resource (which is used to provide processing power for the user-submitted jobs) connected to this Storage/Data Resource by means of TCP/IP network links. Furthermore, each MicroGrid site can have a Computational Resource farm, offering processing power to computationally intensive tasks (e.g. rendering tasks). Different MicroGrid network topologies can automatically be generated by MediaNSG: for now point-to-point topologies, in which clients are directly connected to the Storage/Data Resources, and ring topologies are supported. Other network topologies can easily be added manually or one can use GridG [38], a tool to generate realistic Grid topologies (which is supported by the Grid simulation core). Also, additional Storage/Computational resources can be added to the topology to allow simulation of dedicated Storage/Computational server farms (e.g. used for rendering). A multitude of task scheduling algorithms (e.g. network aware scheduling, service aware scheduling, application level scheduling) are available from MediaNSG and can be employed to schedule user submitted tasks on the MicroGrid/MacroGrid resources. Once a suitable media company profile has been selected or constructed, users can automatically generate a Tcl script describing the scenario's topology, together with the different company profiles to MediaNSG. The MediaNSG simulator itself has been *gridified* in that it is able to run in a Grid environment (all MediaNSG simulations described in Section 7 were run on an LCG-2.6.0 Grid [39] comprised dual Opteron 242 1.6 GHz worknodes with 2 GB RAM per cpu, and operating under Scientific Linux 3).

7. Sample simulation results

In what follows, MediaNSG is used to construct realistic MediaGrid topologies and simulate some *proof-of-concept* MediaGrid situations. In all simulations presented here, each user was associated with a Computational Resource, with all Computational Resources having equal reference processing

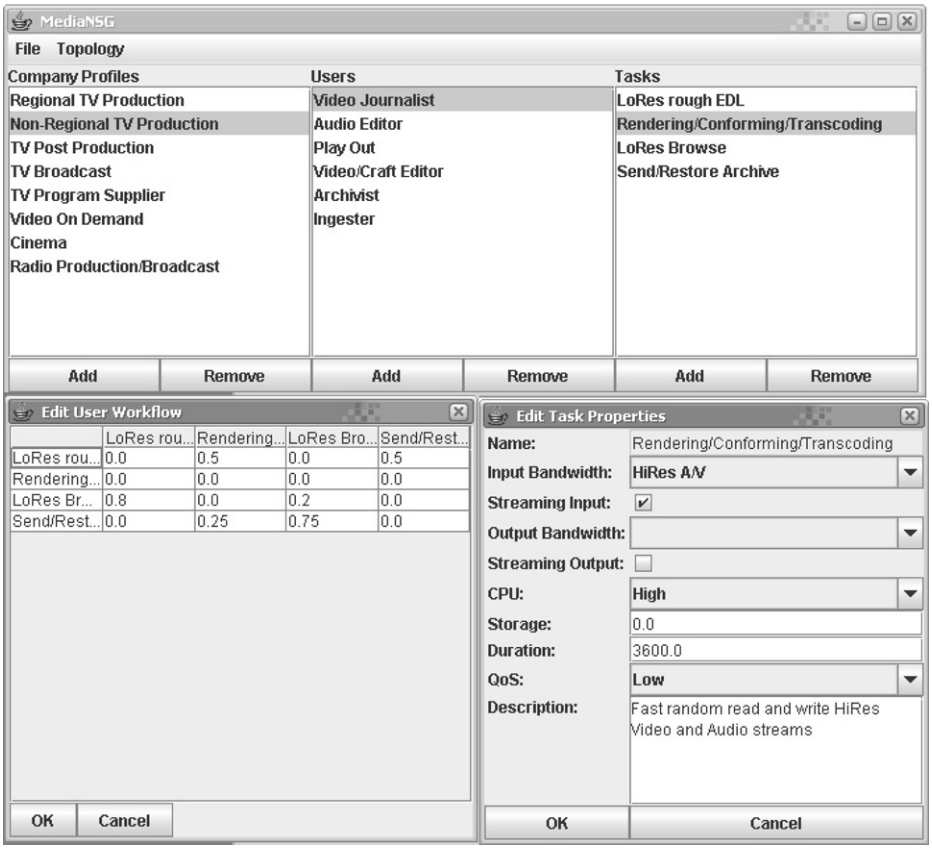


Fig. 14. MediaNSG frontend.

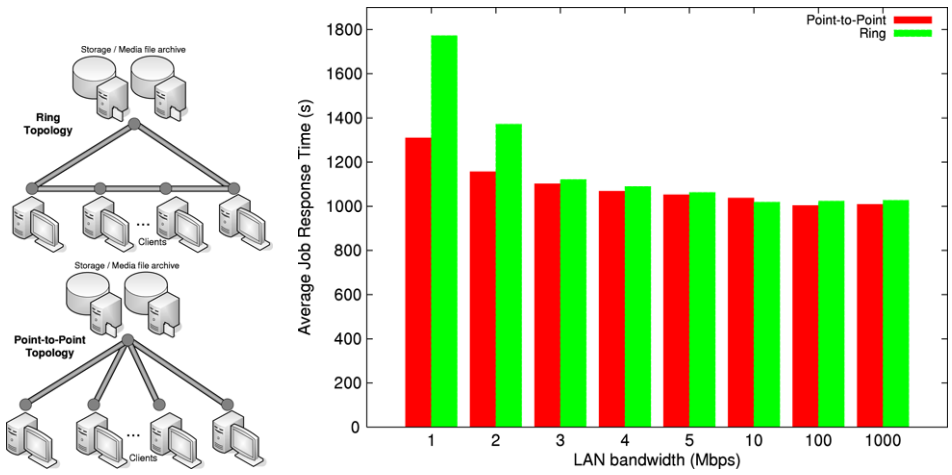
capabilities. We limited Storage Resource/Data Resource access to a maximum read/write throughput of 5600 Mbps (which is realistic as a proof-of-concept storage element array interconnected by fiber channel technology and attaining these speeds is deployed in the FIPA project). User tasks with a duration of 100% were mapped to a simulated duration of 3600 s (e.g. from Fig. 13 we can see that each Quality Checking/HiRes browse task by a producer/director takes 1080 s to complete on a reference processing element).

7.1. MicroGrid topology

In a first batch of simulations, we constructed a Television Broadcasting MicroGrid, and parameterized LAN network bandwidth (LAN interconnections are in this case the network connections between the different clients and the central MicroGrid storage/archiving element containing I/O data for the different tasks) from 1 Mbps to 1000 Mbps. We mapped low, medium and high CPU usage to respectively 10%, 50% and 90% processor utilisation on a reference processor in this simulation. In this first case a point-to-point connection between clients and storage element was provided and network aware scheduling (see Section 4.6) was employed. We measured the average job response times (we define job response times as the difference between the time the job ends, and the time it was submitted to the scheduling service) and notice (see Fig. 15) that tasks experience serious delays when MicroGrid LAN bandwidth is less than 2 Mbps (due to the fact that data input/output retrieval/storage is stalled by network congestion, thereby stalling computational progress). It is also interesting to note that the difference between 10 Mbps and 1 Gbps interconnections is relatively small (on an average tasks took 29 s longer to run when 10 Mbps network technology was used).

In the next batch of simulations, we changed the point-to-point network connection from clients to the storage/archiving element to a ring topology. The results show that at least 3 Mbps interconnections are needed if tasks are to be executed without substantial delays. This can be explained because of the ring configuration of the network topology, network congestion on one link will influence more than one client's job response times (as opposed to the point-to-point network). It is interesting to note that when network bandwidths of 5 Mbps or more are available, the difference between point-to-point connections and ring connections is virtually non-existent, while a ring topology has the benefit of protecting client machines from single link failures.

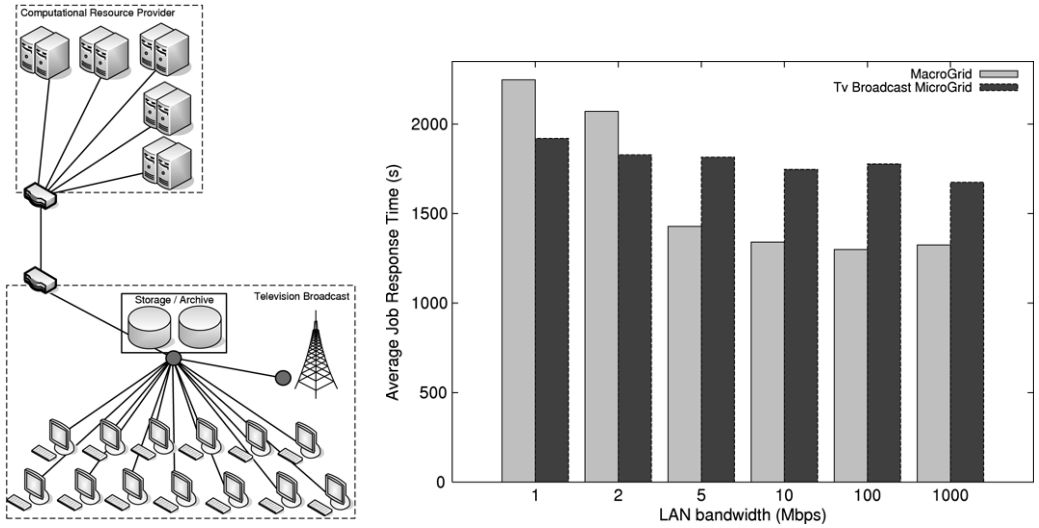
The average job hop count (number of hops storing job output data from Computational to Storage Resource, plus the number of hops for retrieving job input data from Data Resource to Computational Element) when a ring topology was used, was 7.88, while hop count when using point-to-point connections was 2.



(a) MicroGrid ring Vs. point-to-point.

(b) Influence of topology on avg. job response time.

Fig. 15. Influence of network topology on avg. job response time in TV Broadcasting company.



(a) Simulated topology.

(b) MacroGrid resource exchange.

Fig. 16. Influence of MacroGrid resource usage on avg. job response time.

7.2. *MacroGrid resource sharing*

In this simulation, we connected the Television Broadcasting MicroGrid to a remote Computational Resource provider, offering 5 additional processing elements capable of running at twice the reference processor speed (point-to-point connected to a gateway). We again parameterized MicroGrid LAN bandwidth (and used a point-to-point topology), and low, medium and high task processing requirements in this simulation were respectively set at 20%, 100% and 180% of a reference processor. Jobs running at 180% can be processed on the Television Broadcasting MicroGrid, but will run at 0.55% of their normal speed. The link connecting the remote resource provider to the Television Broadcasting MicroGrid is a dedicated link, and in each simulation, it was given the same bandwidth as the Television Broadcast company's LAN bandwidth. The television broadcast's scheduling service (utilising a non-network aware scheduling algorithm as described in Section 4.6) queried both its own Information Service and the Computational Resource provider's Information Services for resources adhering to the job's requirements, and both returned status information regarding the provided resources.

From Fig. 16 we can see that the average task response times drop significantly when the Television Broadcast MicroGrid is given access to the Computational Resource provider's assets *if the interconnecting link bandwidth does not go below* 5 Mbps. Indeed, since network unaware scheduling is used, the scheduler looks at the state of available Computational and Storage Resources, but does not take into account the state of the network links interconnecting these resources. If the available link bandwidth between MicroGrid sites drops, it

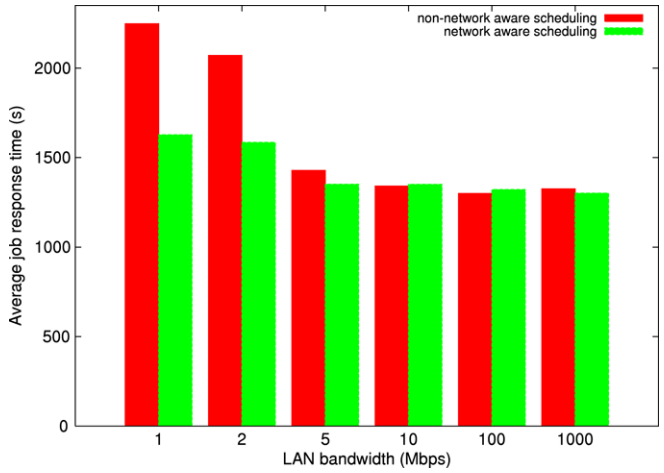


Fig. 17. Non-network aware versus network aware scheduling.

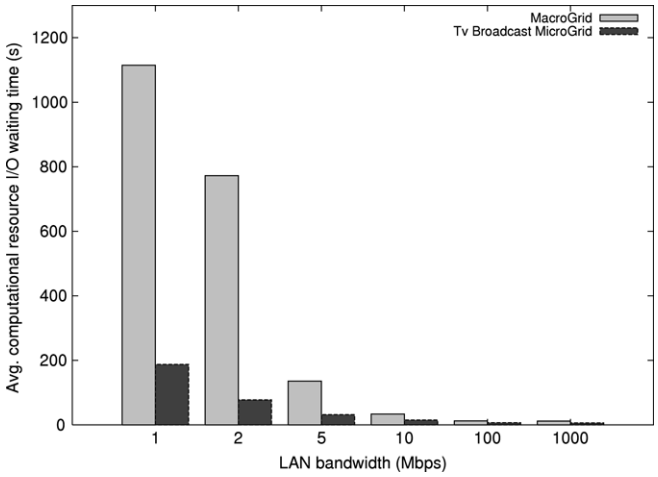


Fig. 18. Avg. time a reserved computational resource spends waiting on I/O.

can be detrimental to schedule jobs for processing on remote resources (as can be seen on Fig. 16 for bandwidths of 1 and 2 Mbps), since the network links connecting these resources to the job's originating site's storage element will become a bottleneck.

If the network did not hamper computational processing (5 Mbps or more interconnections), average job response times were up to 30% better when MacroGrid's resource usage services were being used (in this case allowing migration of jobs from the TV broadcast company to the Computational Resource provider).

7.3. Network aware vs. non-network aware scheduling

To overcome the problem described in the previous simulation, a network aware scheduling algorithm needs to be employed. We simulate the same MacroGrid topology as in Section 7.2 and compare average job response times when scheduling jobs by using network aware on one hand and non-network aware scheduling algorithms on the other. From Fig. 17 we see that at low bandwidths, the average job response time no longer exhibits bad performance when using a network aware scheduling algorithm. This is due to the fact that the network aware algorithm will schedule jobs for processing on remote resources only if the network links connecting these resources to the required storage resource/archive support transferring the job's I/O data at sufficient speeds so as not to waste reserved processing time.

7.4. Resource usage efficiency

From the simulations performed in Section 7.2 we calculated the average amount of time during which a Computational Resource was reserved for a job without being able to continue processing (i.e. 'idling'), because job processing was stalled while waiting for necessary input/output data to be received/sent. From Fig. 18 we can see that utilising remote resources (the MacroGrid case) is not efficient if available connection bandwidth drops below 5 Mbps. At 1 Mbps local MicroGrid task processing capabilities are being hindered by network bottlenecks towards the local storage element/archive.

8. Conclusion

Since media production/broadcast companies are more and more evolving to file based media handling instead of tape based, and since these companies tend to have high requirements regarding Quality of Service, the need to integrate state-of-the-art IT technology in this domain is becoming mandatory. A second important evolution in this field is the need for collaboration amongst media companies, not only on application level, but also to share resources and media repositories. Grid computing can offer a solution in this case, with support for resource/data sharing and advanced collaborative virtual organizations crossing media company boundaries. We therefore propose the use of a MediaGrid architecture consisting of MicroGrid and MacroGrid constituents. The development of suitable scheduling and service management algorithms in a MediaGrid context, can only happen thoroughly if multiple collaboration scenarios are investigated. Setting up a physical MediaGrid testbed for evaluation purposes is cumbersome and time consuming though, so we need to resort to MediaGrid simulation. To this end, we have developed MediaNSG, a digital media company specific Grid simulator built on top of the ns-2 network simulator, and capable of accurately simulating a variety of MediaGrid setups. The MediaNSG simulation environment has been explained in detail in this paper, along with typical MediaGrid profiles for applications, users and digital media companies. Finally, through a set of proof-of-concept simulations, we have shown the added value of our simulator framework. In the (near) future, we will employ MediaNSG scheduling and management algorithm evaluation results in the implementation of the Geisha (Grid Enabled Infrastructure for Service Oriented High Definition Media Applications) project.

Acknowledgments

BV and MDL would like to thank the Institute for the Promotion and Innovation through Science and Technology in Flanders (IWT-Vlaanderen). PT and FDT are research assistant and postdoctoral fellow, respectively, funded by the Research Foundation—Flanders (FWO-Vlaanderen). We would also like to thank Luk Overmeire from the VRT and Lode Pattyn, Marc Shulman and Emmanuel Charlet from the VideoPromotion group for supplying information regarding company, user and task profiles. The work contained in this paper is based on and an extension to some of the research works performed in the FIPA project. The FIPA project [10] (File based Integrated Production Architecture), is an IBBT (Interdisciplinary institute for BroadBand Technology [11]) project aiming at the development of an IP based architecture to share storage and computing power on single or multiple sites. Application areas are digital media production, e-security, e-health, etc.

References

[1] S. Gilheany, Projecting the cost of magnetic disk storage over the next 10 years. http://www.archivebuilders.com/whitepapers/22011p.pdf, 2001.

[2] Mass storage news—Opportunities and challenges in data storage and retrieval. http://www.nexsan.com/pdfs/MassStorageNews_6_02.pdf, 2002.

[3] COTS Technology Focus Data Collection, Paradigm shift hits digital media storage. http://www.cotsjournalonline.com/pdfs/2003/05/cots05_techfocus2.pdf, 2003.

[4] EBU project group P/meta metadata exchange scheme, V. 1.0. http://www.ebu.ch/en/technical/trev/trev_290-hopper.html.

[5] The Globus alliance. http://www.globus.org/.

[6] T.J. Harmer, et al., GridCast—Using the grid in broadcast infrastructures, in: Proceedings of UK e-Science All Hands Meeting, AHM03, 2003.

[7] T.J. Harmer, et al., GridCast—A service architecture for the broadcasting media, in: Proceedings of UK e-Science All Hands Meeting, AHM04, 2004.

[8] VRT—The Flemish Radio- and Television Network. http://www.vrt.be/vrt_master/vrt_en_homepage/index.shtml.

[9] MediaGrid.org consortium. http://mediagrid.org/.

[10] FIPA—file based integrated production architecture project. https://projects.ibbt.be/fipa/.

[11] Institute for BroadBand Technology, http://www.ibbt.be.

[12] GEISHA—Grid Enabled Infrastructure for Service Oriented High Definition Media Applications. https://projects.ibbt.be/geisha/.

[13] Force10 networks white paper—building scalable digital media post production networks: The role of ethernet. http://www.force10networks.com.

[14] SGI white paper—broadcast media management in a data-centric workflow. http://www.sgi.com.

[15] Anystream Agility. http://www.anystream.com/.

[16] IBM Digital Media Center. http://www-03.ibm.com/industries/media/doc/content/solution/1394111111.html.

[17] Omneon MediaGrid. http://www.omneon.com/mediagrid.

[18] Open Grid services architecture use cases. http://www.gridforum.org/documents/GWD-I-E/GFD-I.029v2.pdf.

[19] The network simulator—NS2. http://www.isi.edu/nsnam/ns.

[20] Jason Liu, L. Felipe Perrone, David M. Nicol, Michael Liljenstam, Chip Elliott, David Pearson, Simulation modeling of large-scale ad-hoc sensor networks, in: European Simulation Interoperability Workshop, 2001.

[21] A. Varga, OMNeT++, IEEE Network Interactive 16 (4) (2002).

[22] Real time streaming protocol—RFC2326. http://tools.ietf.org/html/rfc2326.

[23] Real-time transport protocol and RTP control protocol—RFC3550. http://tools.ietf.org/html/rfc3550.

[24] Resource ReSerVation protocol—RFC2205. http://tools.ietf.org/html/rfc2205.

[25] Atsuko Takefusa, Satoshi Matsuoka, Henri Casanova, Francine Berman, A study of deadline scheduling for client-server systems on the computational grid, in: HPDC '01: Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing, HPDC-10'01, 2001.

[26] A. Takefusa, O. Tatebe, S. Matsuoka, Y. Morita, Performance analysis of scheduling and replication algorithms on grid datafarm architecture for high-energy physics applications, in: Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing, HPDC-12, 2003.

[27] Xin Liu, Huaxia Xia, Andrew Chien, Validating and scaling the microgrid: A scientific instrument for grid dynamics, Journal of Grid Computing 2 (2004) 141–161.

[28] Arnaud Legrand, Loris Marchal, Henri Casanova, Scheduling distributed applications: The SimGrid simulation framework, in: CCGRID'03: Proceedings of the 3st International Symposium on Cluster Computing and the Grid, 2003.

[29] J. Lerouge, A. Legrand, MetaSimGrid: Towards realistic scheduling simulation of distributed applications, ENS-LIP Research Report 2002-28, 2002.

[30] Anthony Sulistio, Gokul Poduvaly, Rajkumar Buyya, Chen-Khong Tham, Constructing a Grid simulation with differentiated network service using GridSim, in: Proc. of the 6th International Conference on Internet Computing, ICOMP'05, 2005.

[31] Hung-Ying Tyan, Chao-Ju Hou, JavaSim: A component-based compositional network simulation environment, in: Proc. of Western Simulation Multiconference—Communication Networks and Distributed Systems Modeling and Simulation, 2001.

[32] John A. Miller, Andrew F. Seila, Xuewei Xiang, The JSIM web-based simulation environment (Web-Based Modeling and Simulation), Future Generation Computer Systems 17 (2000) 119–133 (special issue).

[33] K. Ranganathan, I. Foster, Identifying dynamic replication strategies for a high performance Data Grid, in: Proc. of the International Grid Computing Workshop, 2001.

[34] Parsec: Parallel simulation environment for complex systems. http://pcl.cs.ucla.edu/projects/parsec.

[35] David G. Cameron, Ruben Carvajal-Schiaffino, A. Paul Millar, Caitriana Nicholson, Kurt Stockinger, Floriano Zini, Evaluating scheduling and replica optimisation strategies in OptorSim, in: 4th International Workshop on Grid Computing, Grid2003, 2003.

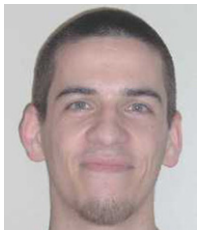
[36] The DataGrid project. http://eu-datagrid.web.cern.ch/eu-datagrid/.

[37] Video promotion. http://www.videopromotion.be/.

[38] P. Dinda, D. Lu, GridG: generating realistic computational Grids, Performance Evaluation Review 30 (4) (2003).

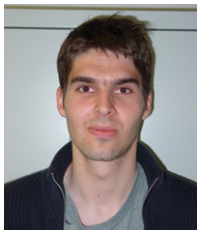
[39] LHC computing Grid project. http://lcg.web.cern.ch/LCG.

Bruno Volckaert received his Masters degree in Computer Science from the Ghent University, Belgium, in June 2001. In June 2006, he obtained his Ph.D. degree for his thesis "Architectures and Algorithms for Network and Service Aware Resource Management" from the same university. He is now a research project manager affiliated to the Department of Information Technology of the Ghent University. His main research interests include Grid Management Architectural designs, Grid simulation and Service-Oriented Architectures.



Tim Wauters received his M.Sc. degree in Electrotechnical engineering (option communication techniques) in 2001 from the University of Ghent, Belgium. Since September 2001, he has been working on the design of content distribution and peer-to-peer networks in the Department of Information Technology (INTEC), at the same university. His work has been published in several journals and scientific publications on international conferences.

Marc De Leenheer received his M.Sc. degree in Computer Science Engineering from the Ghent University, Belgium, in June 2003. He is now a research assistant and a Ph.D. student affiliated to the Department of Information Technology of the Ghent University. His main research interests include modeling and optimizing Grid management architectures.



Pieter Thysebaert received his M.Sc. degree in Computer Science Engineering from the Ghent University, Belgium, in June 2001. He is now a research assistant and a Ph.D. student affiliated to the Department of Information Technology of the Ghent University, and has received a scholarship from the FWO (Fund for Scientific Research—Flanders). His main research interests include Grid simulation and modeling of Grid scheduling problems.

Filip De Turck received his M.Sc. degree in Electronic Engineering from the Ghent University, Belgium, in June 1997. In May 2002, he obtained the Ph.D. degree in Electronic Engineering from the same university. From October 1997 to September 2001, Filip De Turck was research assistant with the Fund for Scientific Research—Flanders, Belgium (F.W.O.-V.). At the moment, he is a professor at the Department of Information Technology of the Ghent University. His main research interests include scalable software architectures for telecommunication network and service management, performance evaluation and optimization of routing, admission control and traffic management in telecommunication systems.

Bart Dhoedt received a degree in Engineering from the Ghent University in 1990. In September 1990, he joined the Department of Information Technology of the Faculty of Applied Sciences, University of Ghent. His research, addressing the use of micro-optics to realize parallel free space optical interconnects, resulted in a Ph.D. degree in 1995. After a 2 year post-doc in opto-electronics, he became a professor at the Faculty of Applied Sciences, Department of Information Technology. Since then, he is responsible for several courses on algorithms, programming and software development. His research interests are software engineering and mobile & wireless communications. His current research addresses software technologies for communication networks, peer-to-peer networks, mobile networks and active networks.

Piet Demeester received the Masters degree in Electrotechnical engineering and the Ph.D degree from the Ghent University, Ghent, Belgium in 1984 and 1988, respectively. In 1992 he started a new research activity on broadband communication networks resulting in the IBCN-group (INTEC Broadband communications network research group). In 1993, he became a professor at the Ghent University where he is responsible for the research and education on communication networks. The research activities cover various communication networks (IP, ATM, SDH, WDM, access, active, mobile), including network planning, network and service management, telecom software, internetworking, network protocols for QoS support, etc.

Piet Demeester is the author of more than 300 publications in the area of network design, optimization and management. He is a member of the editorial board of several international journals and has been a member of several technical program committees (ECOC, OFC, DRCN, ICCCN, IZS).